SEP 09 – 11, 2025
CONTAINER days CONFERENCE

From IDP to AIDP: Evolving Your Platform for the Machine Learning Age

CONTAINERDAYS CONFERENCE 2025

Max Körbächer

The Convergence of Platform Engineering and AI ?

What is an AI-IDP?

AI-**Powered** Platforms

Platforms that use AI **to enhance their own capabilities,** providing intelligent assistance, automation, and optimization for development workflows.

AI-**Enabling** Platforms

Platforms designed to **support the development**, deployment, and management **of AI/ML** workloads, making AI capabilities accessible to developers.

Core Benefits

- Accelerated development cycles
- Improved code quality
- Enhanced developer productivity
- Reduced operational overhead

Bridging the Gap: Developers and AI Engineering Teams

Traditional Developers

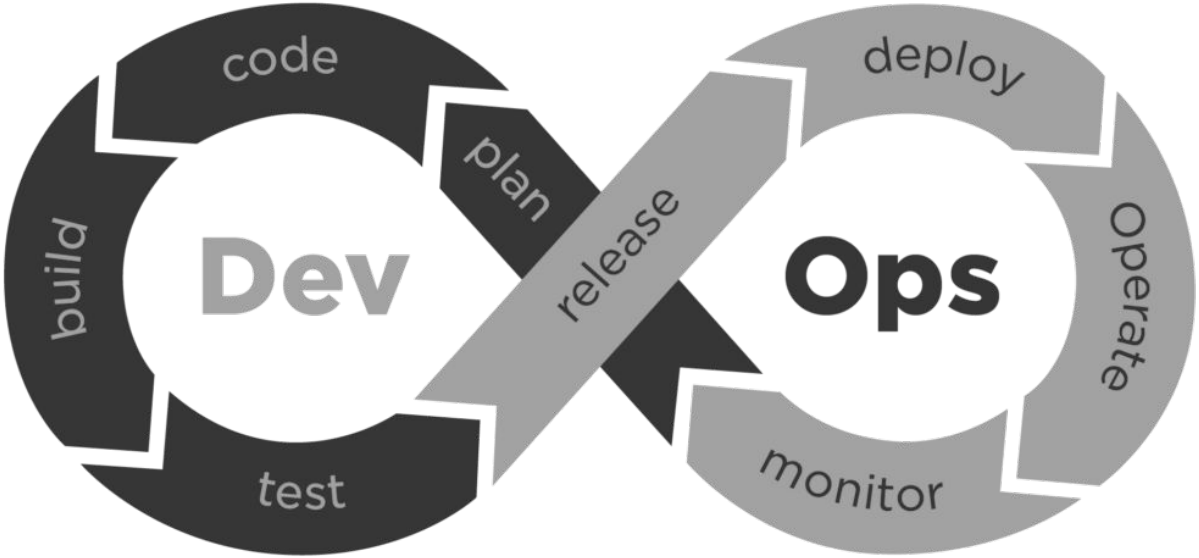
Code-first approach

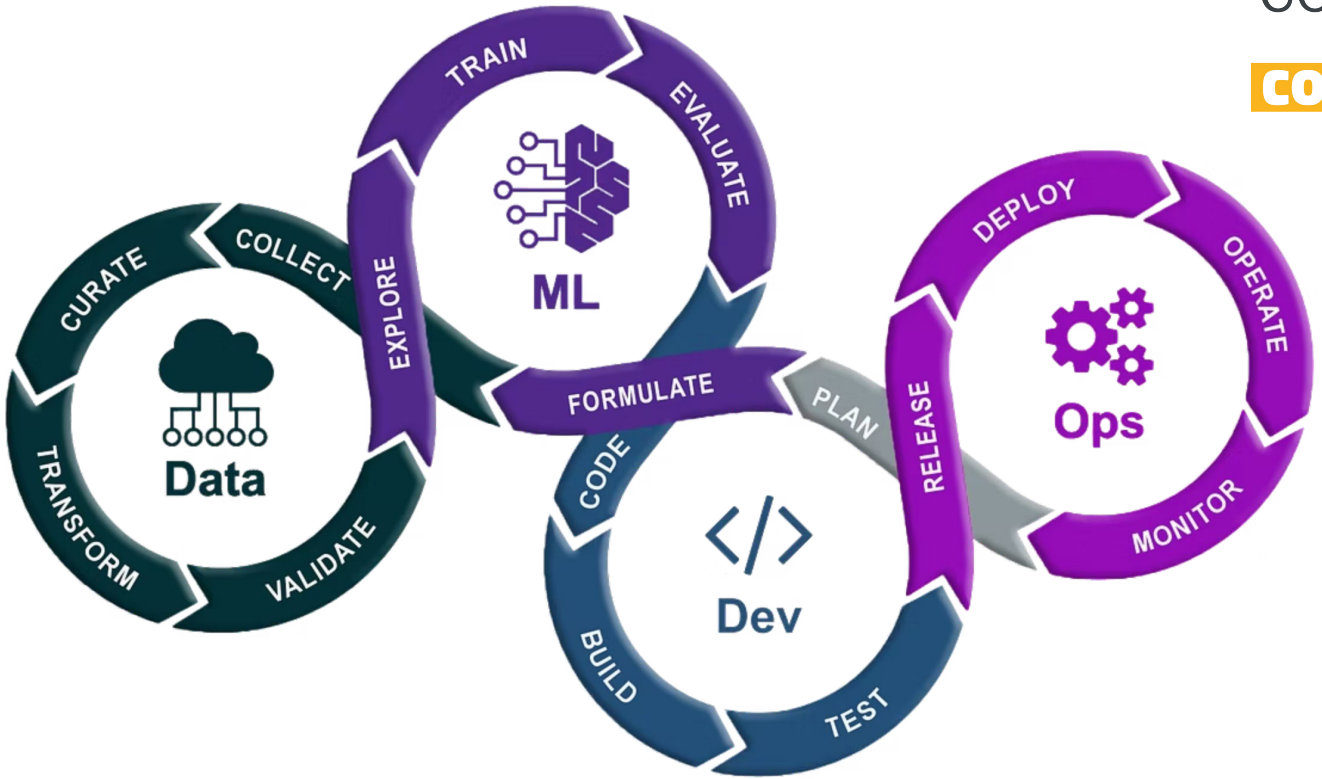
- Small artifacts, predictable CI/CD, auto. testing
- Standard hardware resources
- Scale horizontally

AI/ML Engineers

Experiment-first approach

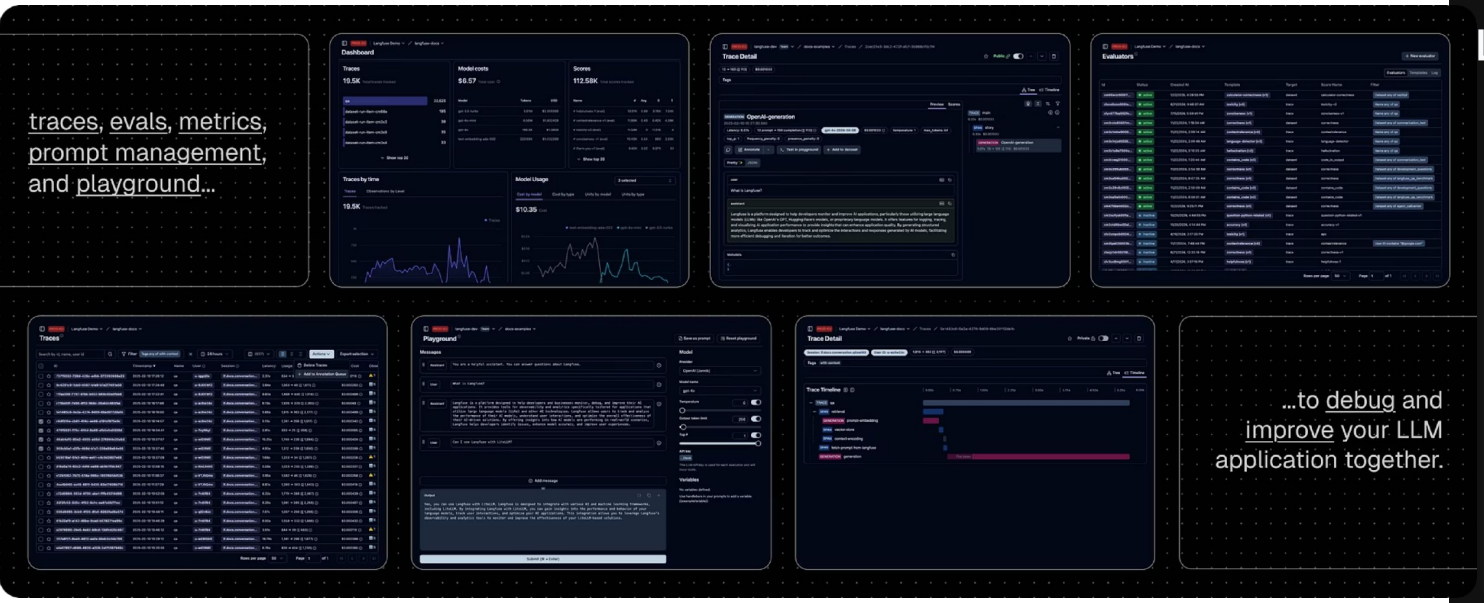
- Iterative experimentation, complex flows
- Large models, massive datasets
- Specialized hardware requirements
- Scale vertically, mix


code
plan
build
Dev
test
release
deploy
Ops
operate
monitor


TRAIN
EVALUATE
ML
EXPLORE
CURATE
COLLECT
Data
TRANSFORM
VALIDATE
FORMULATE
PLAN
CODE
Dev
BUILD
TEST
RELEASE
DEPLOY
OPERATE
Ops
MONITOR

Understanding your users

langfuse as example



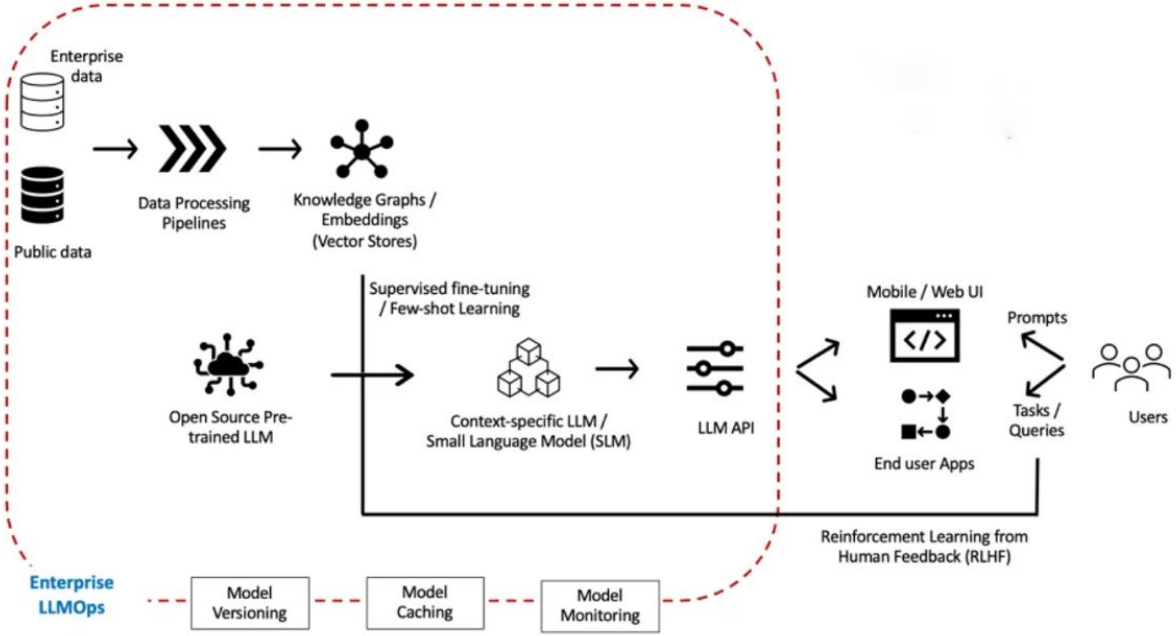
Works with any LLM app and model

SDKs for Python & JS/TS, native integrations for popular libraries and support for OpenTelemetry

Integration overview → Request new integration →

Python SDK	JS/TS SDK	API	OpenTelemetry	OpenAI SDK
LangChain, LangGraph	Llama-Index	CrewAI	LiteLLM	AI SDK
Haystack	Instructor	Semantic Kernel	DSPy	Smolagents
Pydantic AI	AutoGen	Amazon Bedrock	Google Vertex/Gemini	Ollama
Flowise	Langflow	Dify	OpenWeb UI	More

But there is more, right?



Source: Debmalya Biswas

But there is more, right, RIGHT?

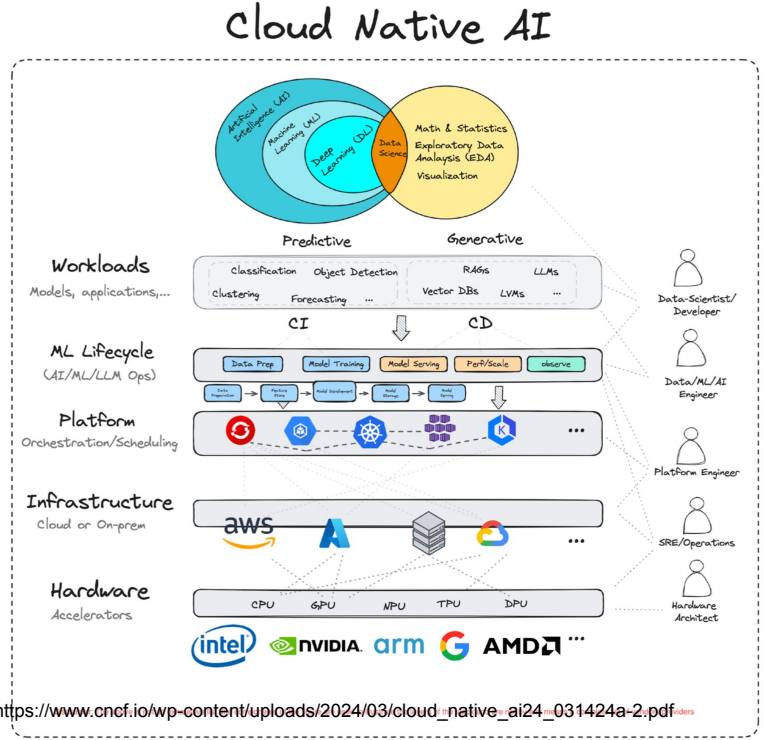


Figure 1
Cloud Native AI

Cloud Native Artificial Intelligence Whitepaper: https://www.cncf.io/wp-content/uploads/2024/03/cloud_native_ai24_031424a-2.pdf

But there is more, right, RIGHT; R3!*gT?

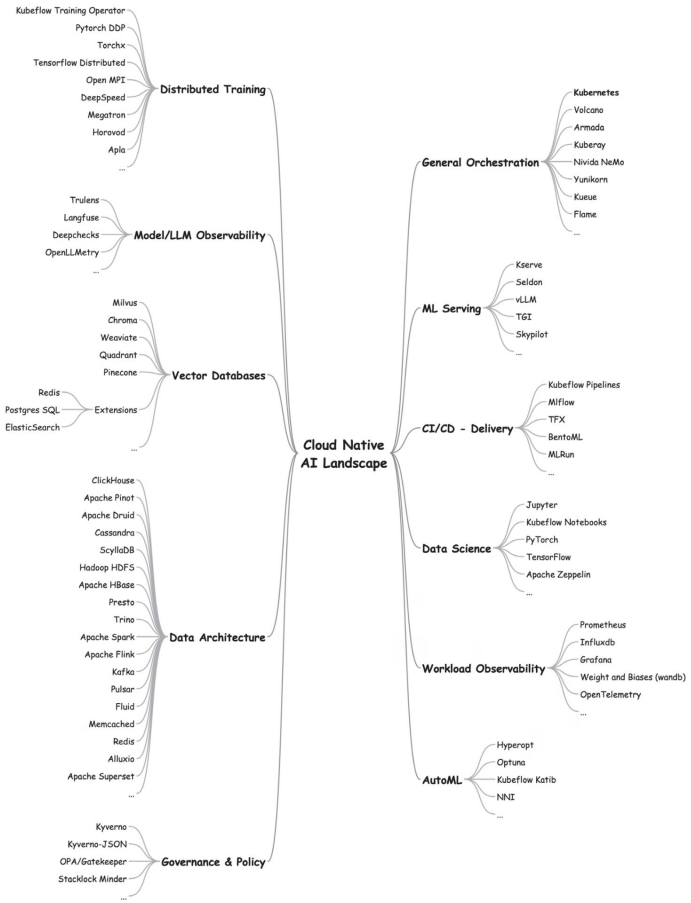
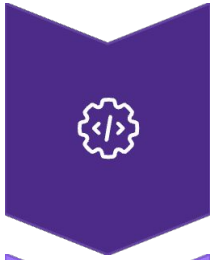


Figure 4
ML Tool to Task Mind Map

Cloud Native Artificial Intelligence Whitepaper: https://www.cn

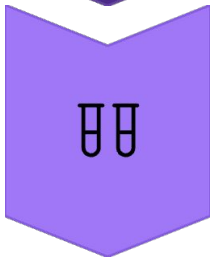
SEP 09 – 11, 2025
CONTAINER days CONFERENCE

The 3 Elements you have to master first



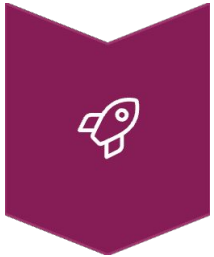
“MLOps” - Pipelines

Orchestrate, manage and scale



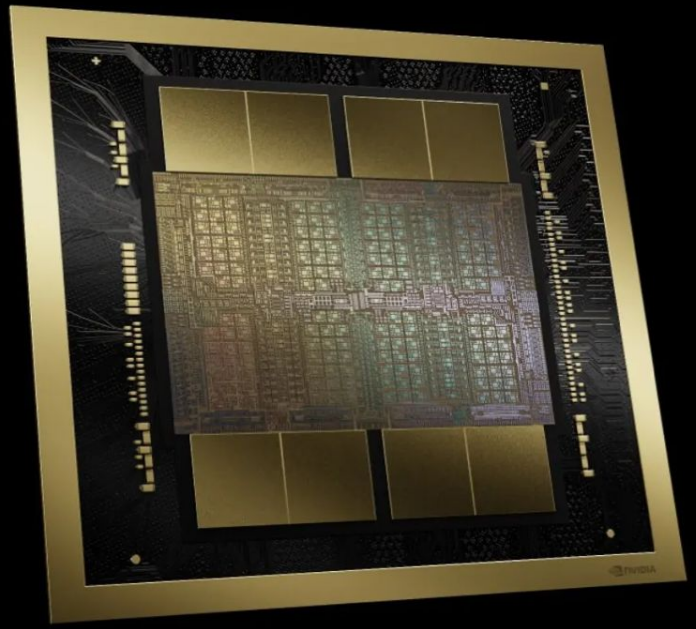
Lifecycle and orchestration of *AI* workload

That’s what we are doing just without AI so far



AI Accelerating Hardware Scheduling & Management

Provide use case oriented hardware



GPU Resource Management

vGPU (Virtual GPU)

Sharing of one GPU with multiple VMs or Containers

MIG (Multi-instance GPU)

Partitioning of a physical GPU for multiple independent workloads

GPU Time-Slicing

Slices GPU time across multiple tasks

CUDA

Software to develop, manage and distribute workload

Physical GPU

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
    - name: gpu-example
      image: nvidia/cuda
      resources:
        limits:
          nvidia.com/gpu: 2
  nodeSelector:
    nvidia.com/gpu.product: A100-PCIE-40GB
    nvidia.com/cuda.runtime: 11.4
    nvidia.com/cuda.driver: 470.161.03
```

MIG

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
    - name: gpu-example
      image: nvidia/cuda
      resources:
        limits:
          nvidia.com/mig-1g.5gb: 1
  nodeSelector:
    nvidia.com/gpu.product: A100-PCIE-40GB
    nvidia.com/cuda.runtime: 11.4
    nvidia.com/cuda.driver: 470.161.03
```

Time Slices with GPU

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
    - name: gpu-example
      image: nvidia/cuda
      resources:
        limits:
          nvidia.com/gpu.shared: 1
  nodeSelector:
    nvidia.com/gpu.product: A100-PCIE-40GB
    nvidia.com/cuda.runtime: 11.4
    nvidia.com/cuda.driver: 470.161.03
```

k8s-device-plugin config file

```yaml
version: v1
sharing:
  timeSlicing:
    resources:
    - name: nvidia.com/gpu
      replicas: 10
    ...
```

Time Slices with MIG

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
    - name: gpu-example
      image: nvidia/cuda
      resources:
        limits:
          nvidia.com/mig-1g.5gb.shared: 1
  nodeSelector:
    nvidia.com/gpu.product: A100-PCIE-40GB
    nvidia.com/cuda.runtime: 11.4
    nvidia.com/cuda.driver: 470.161.03
```

k8s-device-plugin config file

```yaml
version: v1
sharing:
  timeSlicing:
    resources:
    - name: nvidia.com/gpu
      replicas: 10
    - name: nvidia.com/mig-1g.5gb
      replicas: 10
    ...
```

Limitations with this approach

Heterogeneous GPU Support

No native support for integrating more than one GPU type per node, restricting hardware configurations.

Complex Constraint Handling

Lack of robust mechanisms for providing complex constraints when requesting a GPU, making advanced scheduling difficult.

Oversubscription Control

Limited control over how oversubscribed GPUs are shared between jobs, potentially leading to unfair resource distribution.

~~**MPS Integration Challenges**~~

Awkward and overly-burdensome support for Multi-Process Service (MPS), adding operational overhead.

~~**Dynamic MIG Provisioning**~~

Inability to dynamically provision Multi-Instance GPU (MIG) devices based on incoming requests, reducing resource elasticity.

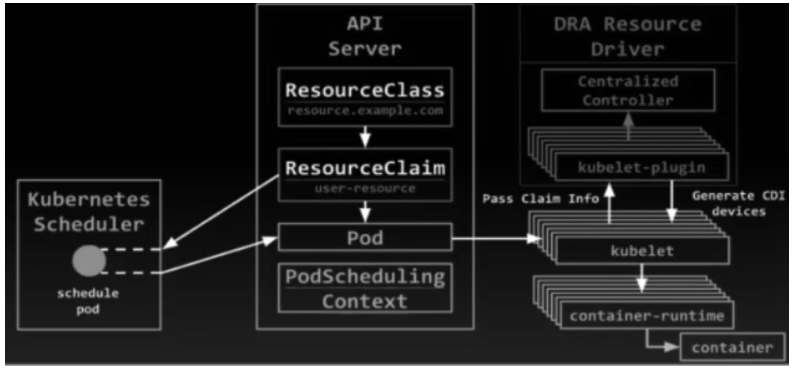
Driver Selection Flexibility

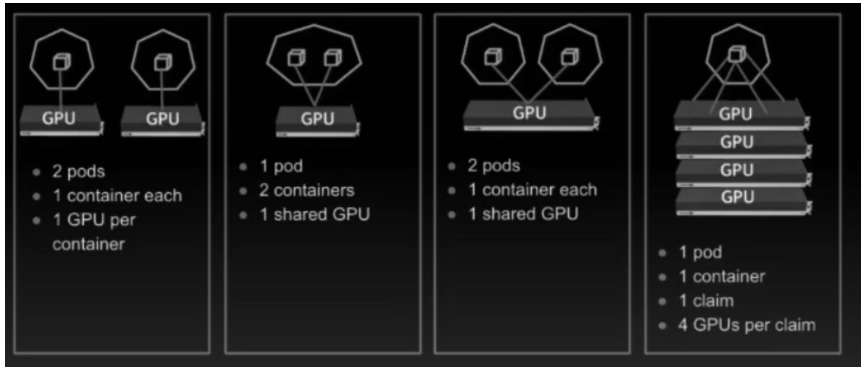
No built-in capability to dynamically choose between NVIDIA and other drivers (e.g., fio) on a per-GPU basis.

DRA - Dynamic Resource Allocation

Allocate **cross-node resources** in Kubernetes

Explicitly **share, partition, and reconfigure** devices **on-the-fly** based on user requests





DRA - Dynamic Resource Allocation

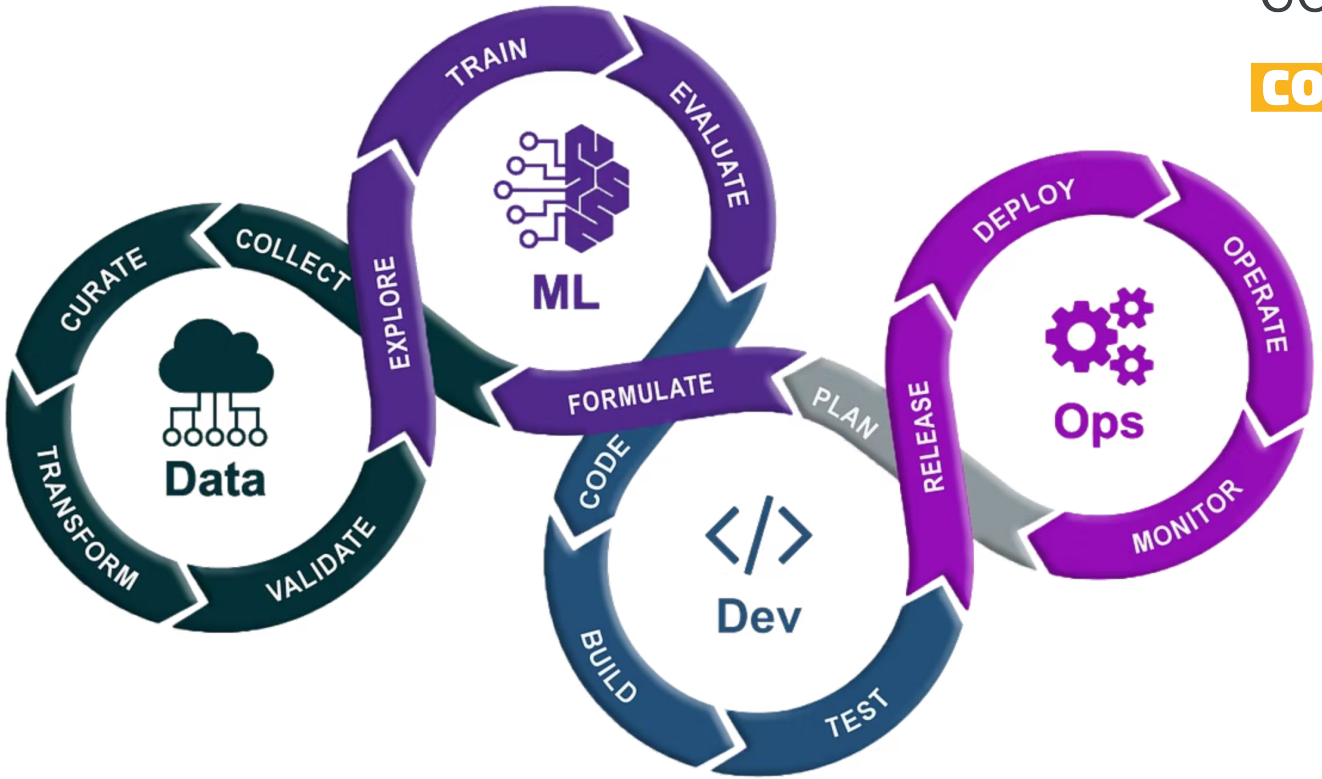
```yaml
---
apiVersion: resource.k8s.io/v1alpha2
kind: ResourceClaimTemplate
metadata:
  name: unique-gpu
spec:
  spec:
    resourceClassName: gpu.nvidia.com

---
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
  - name: ctr
    image: nvidia/cuda
    command: ["nvidia-smi" "-L"]
    resources:
      claims:
      - name: gpu0
      - name: gpu1
  resourceClaims:
  - name: gpu0
    source:
      resourceClaimTemplateName: unique-gpu
  - name: gpu1
    source:
      resourceClaimTemplateName: unique-gpu
```

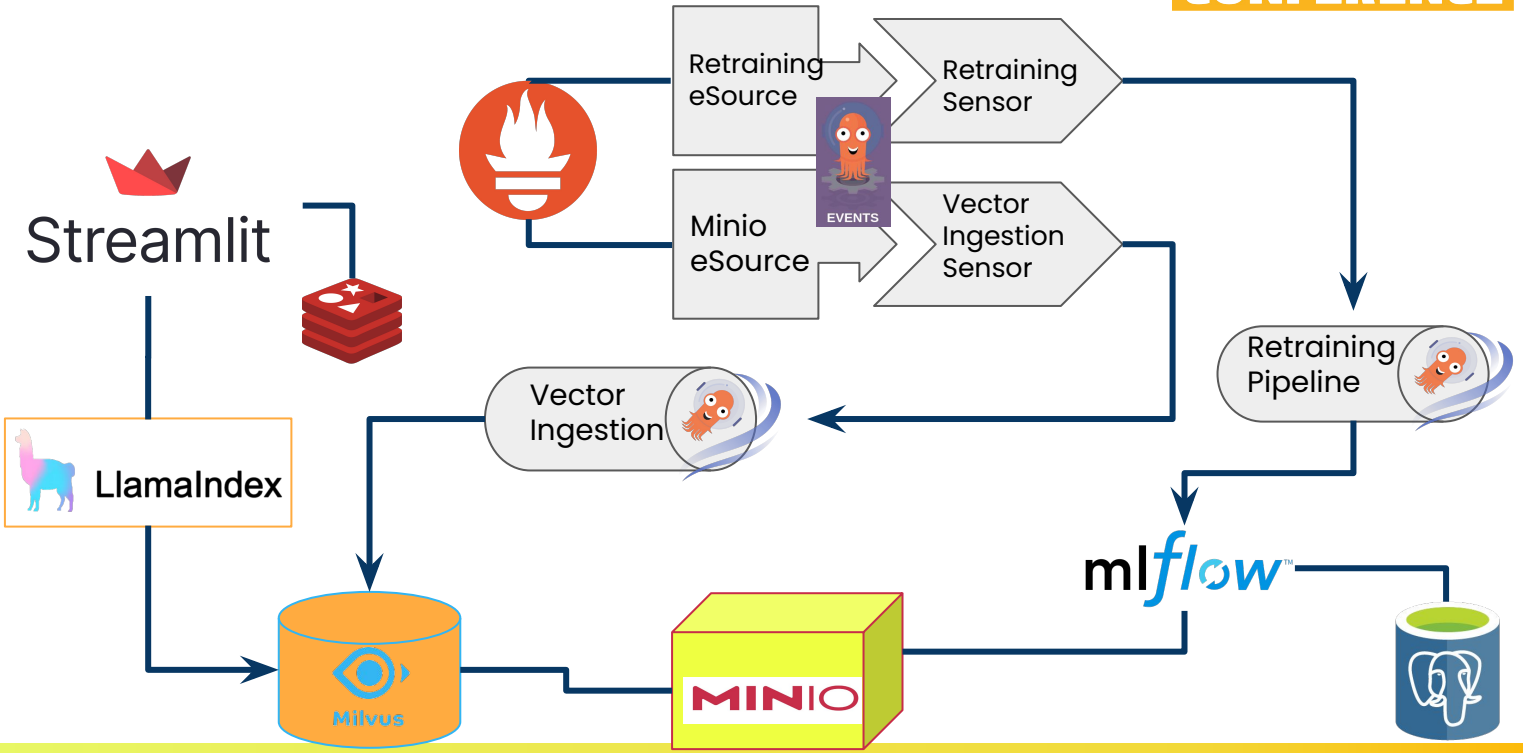
Associated with the DRA Driver and installed by the cluster admin

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
    - name: ctr0
      resources:
        claims:
        - name: gpu
    - name: ctr1
      resources:
        claims:
        - name: gpu
resourceClaims:
  - name: gpu
    source:
      resourceClaimName: unique-gpu
```

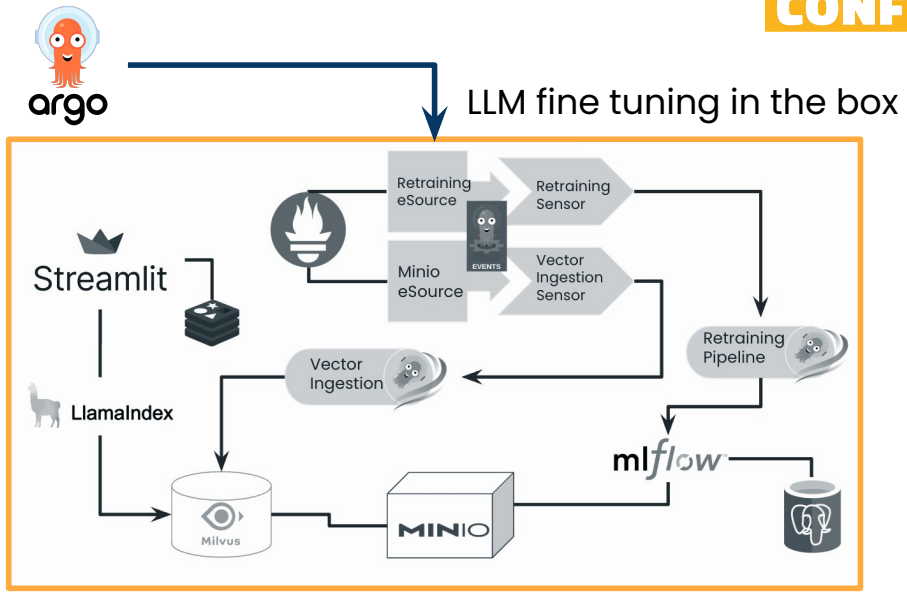
Shared access to same underlying GPU


TRAIN
EVALUATE
ML
EXPLORE
CURATE
COLLECT
Data
TRANSFORM
VALIDATE
FORMULATE
PLAN
CODE
Dev
BUILD
TEST
RELEASE
DEPLOY
OPERATE
Ops
MONITOR

ML/LLMOps example Architecture



ML/LLMOps example Architecture



Kubernetes

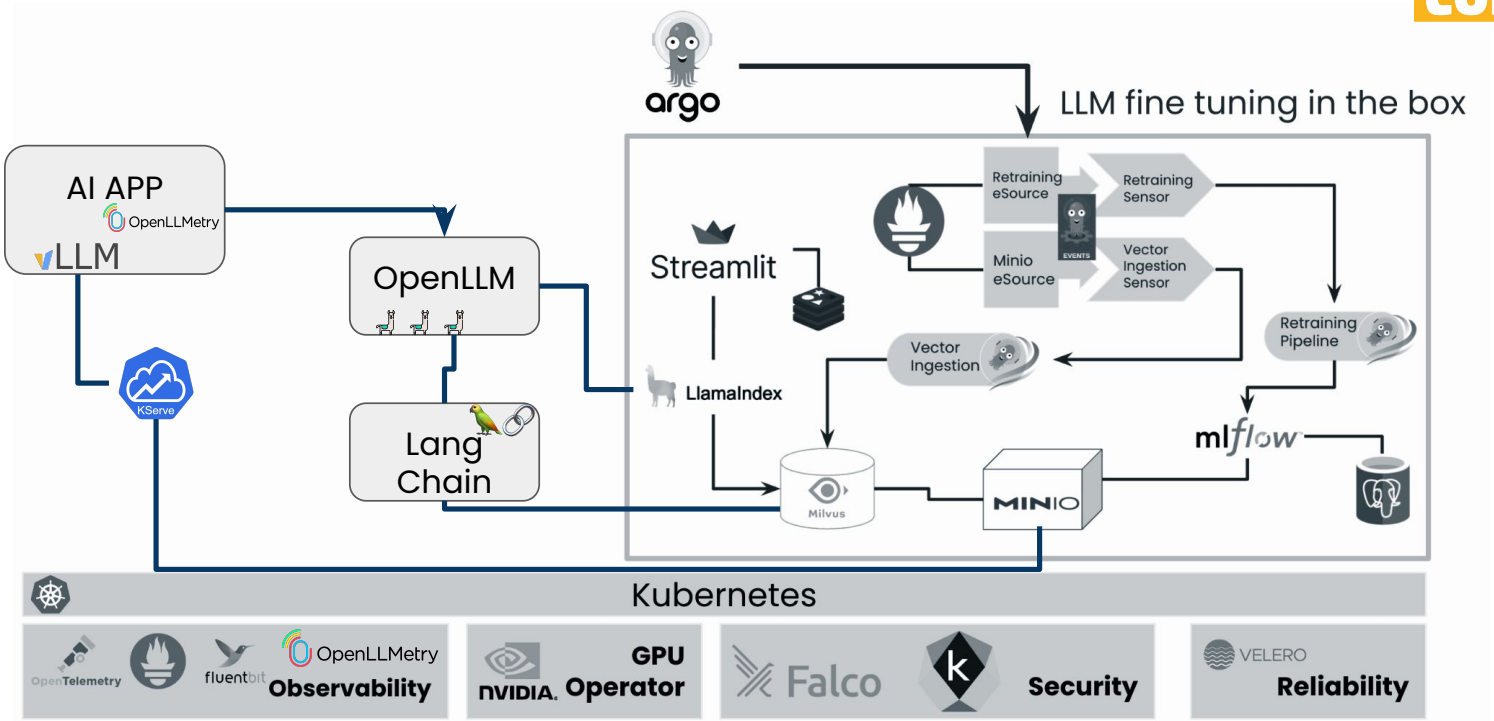
OpenTelemetry fluentbit Observability

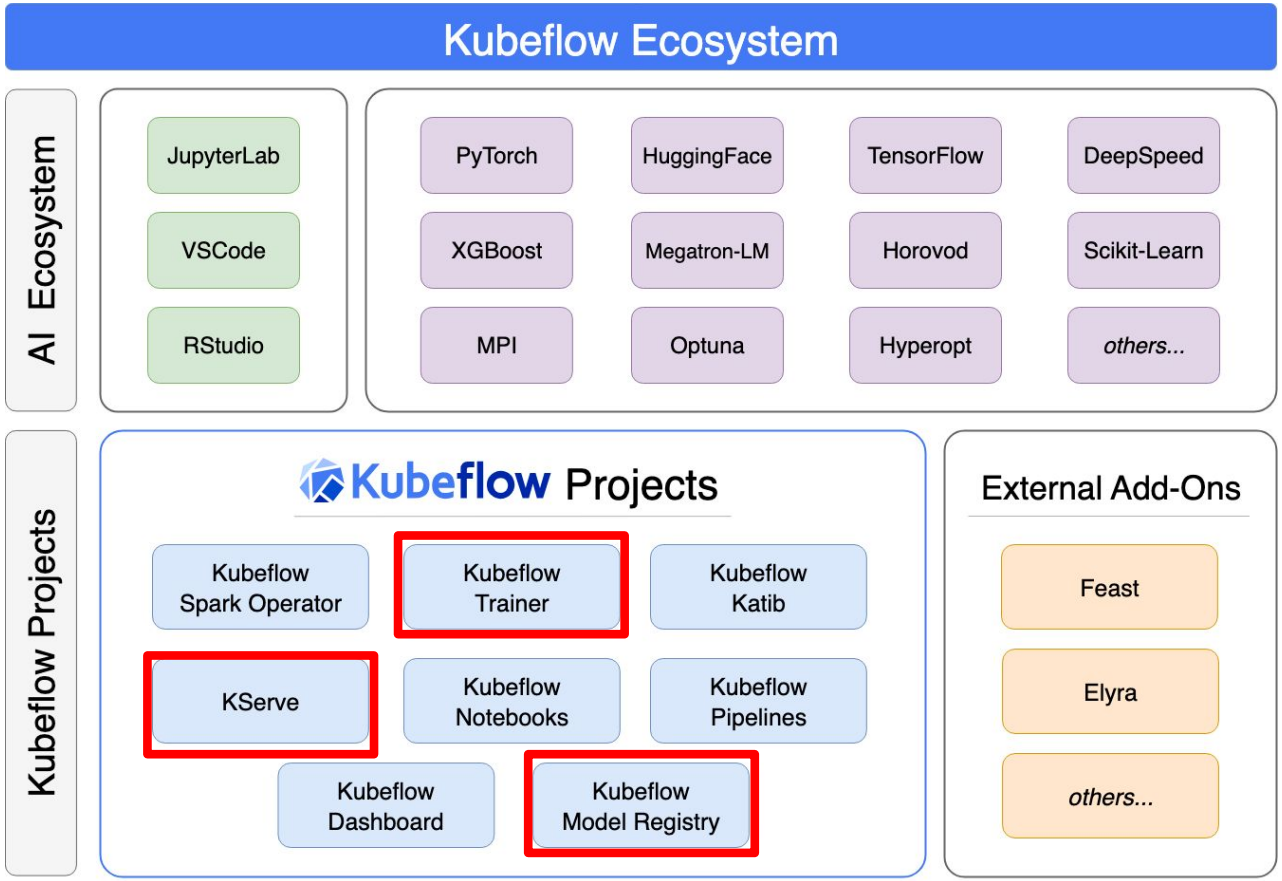
NVIDIA GPU Operator

Falco Security

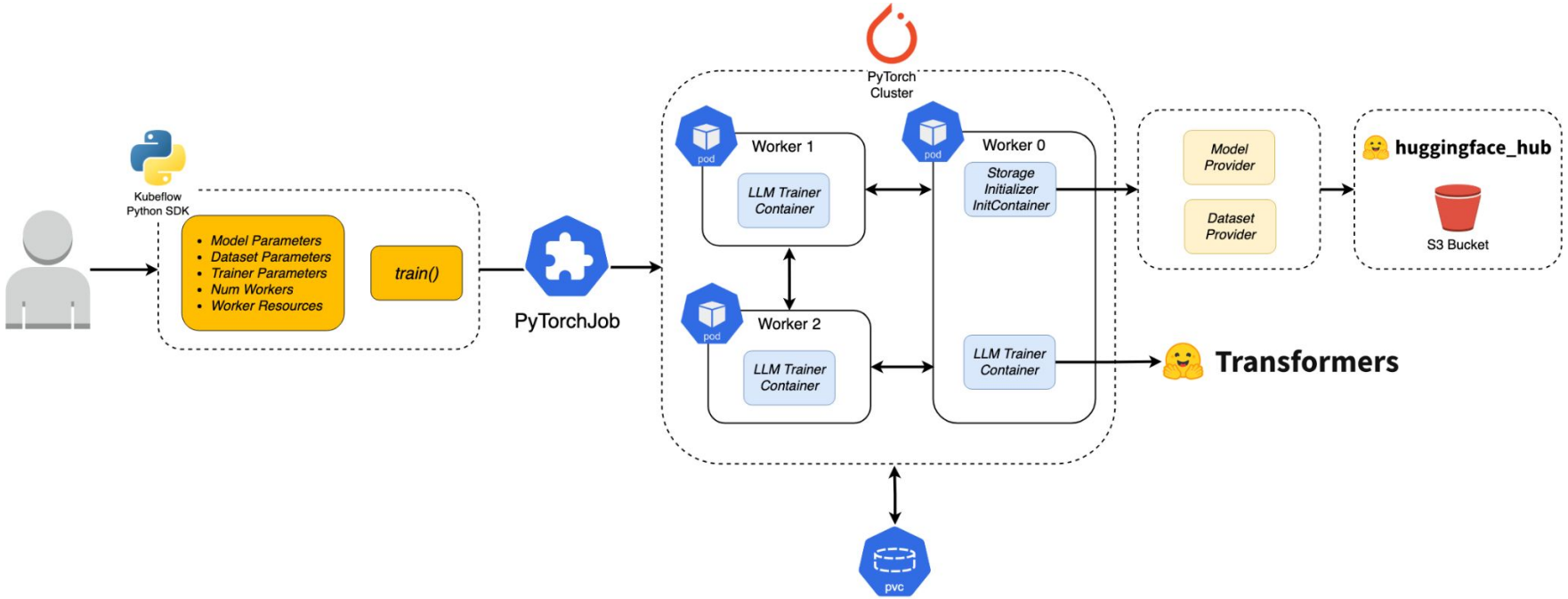
VELERO Reliability

ML/LLMOps example Architecture

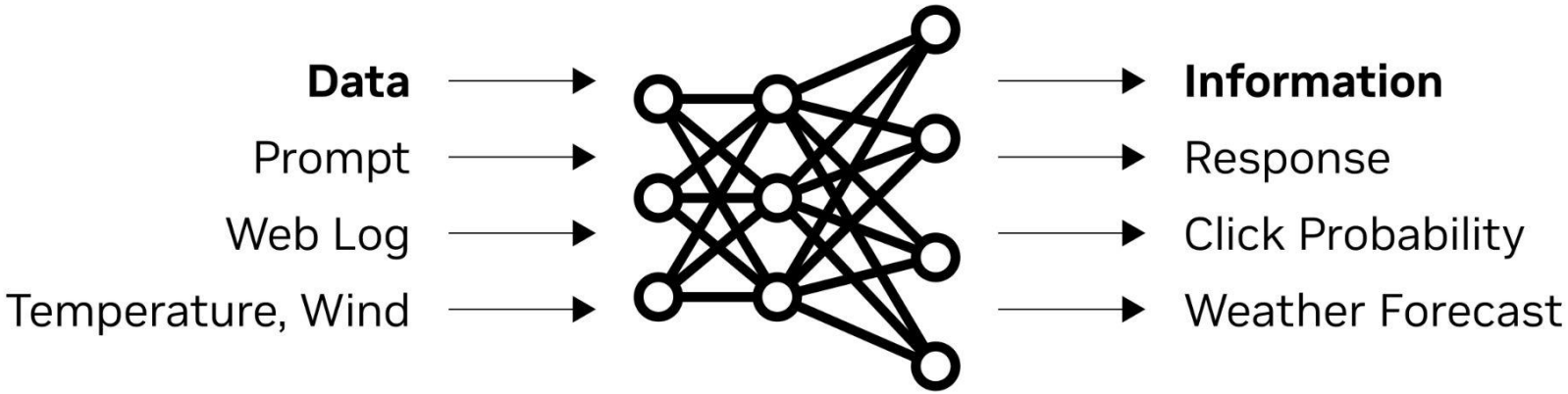


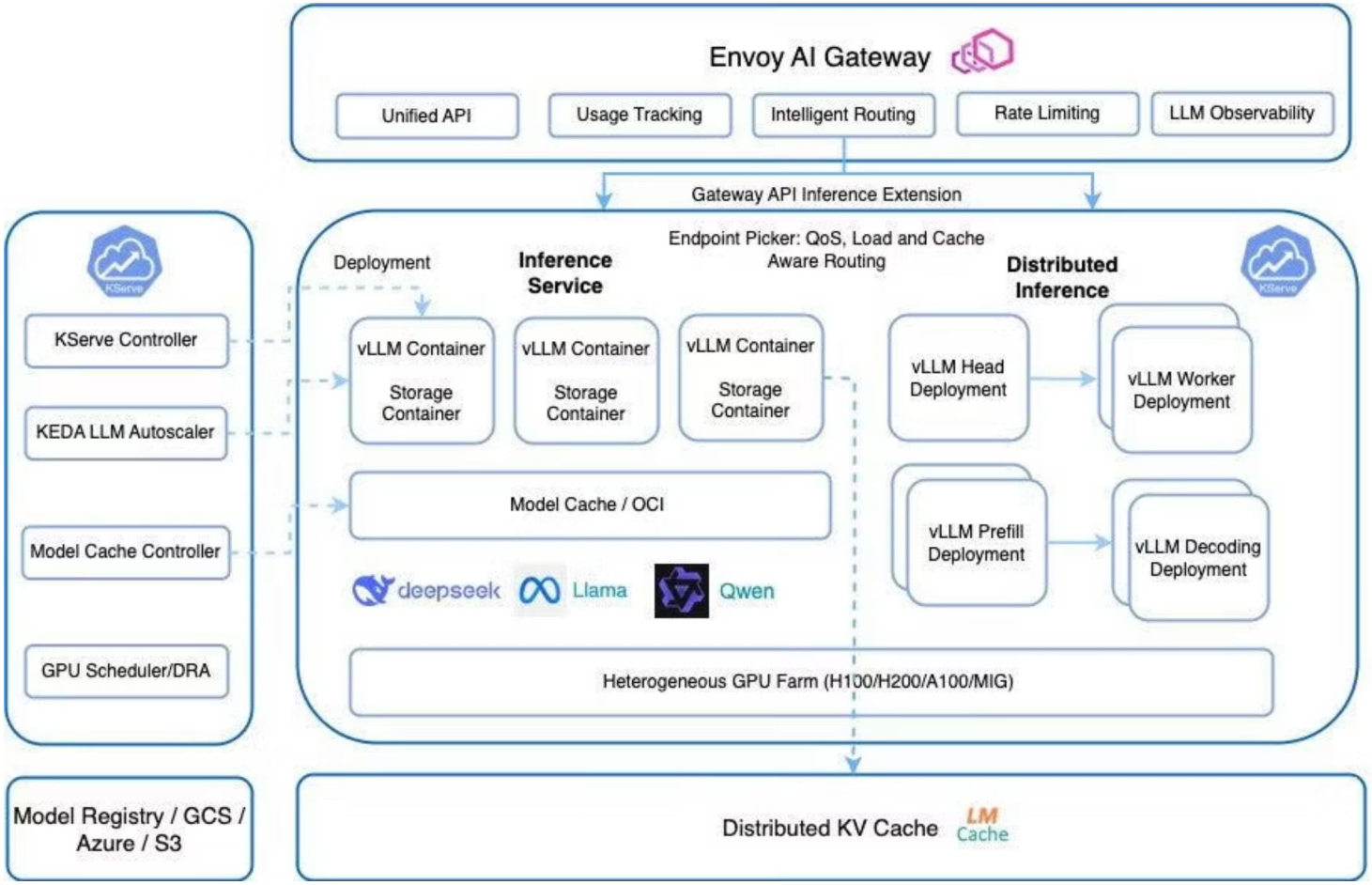

Kubeflow Ecosystem
AI Ecosystem
JupyterLab
VSCode
RStudio
PyTorch
HuggingFace
TensorFlow
DeepSpeed
XGBoost
Megatron-LM
Horovod
Scikit-Learn
MPI
Optuna
Hyperopt
others...
Kubeflow Projects
Kubeflow Projects
Kubeflow Spark Operator
Kubeflow Trainer
Kubeflow Katib
KServe
Kubeflow Notebooks
Kubeflow Pipelines
Kubeflow Dashboard
Kubeflow Model Registry
External Add-Ons
Feast
Elyra
others...

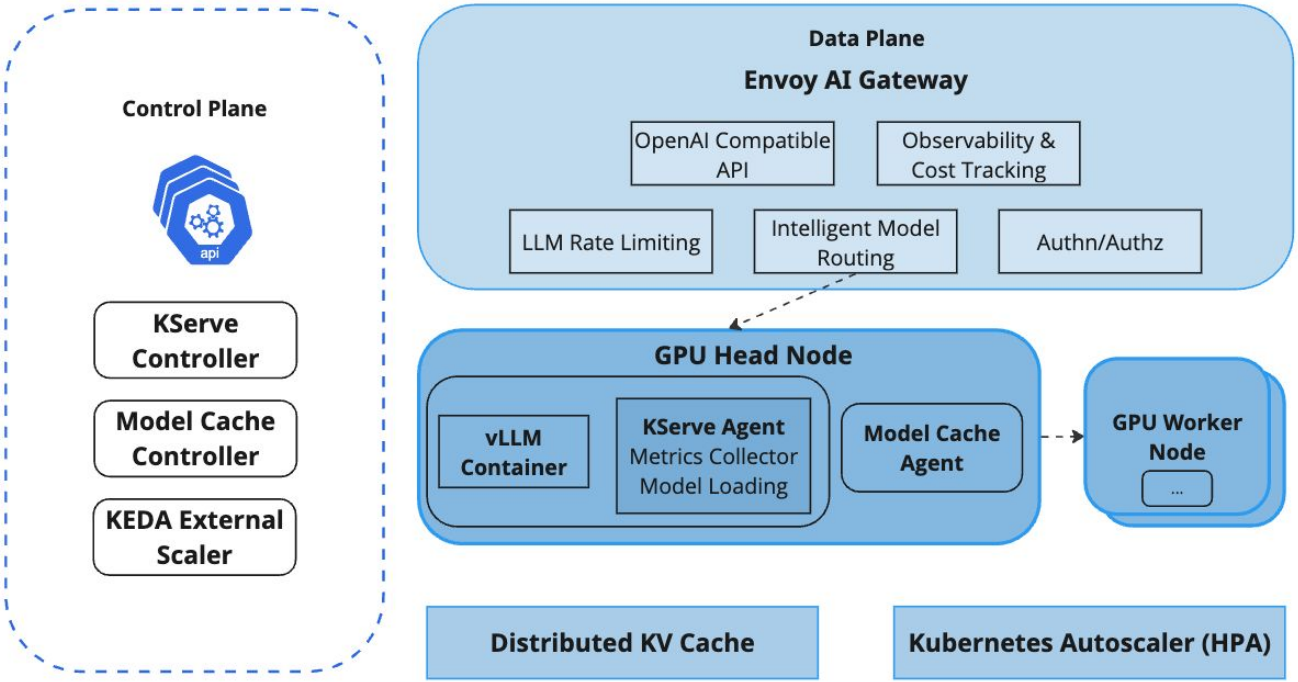
Fine-tuning with kflow



General Inference



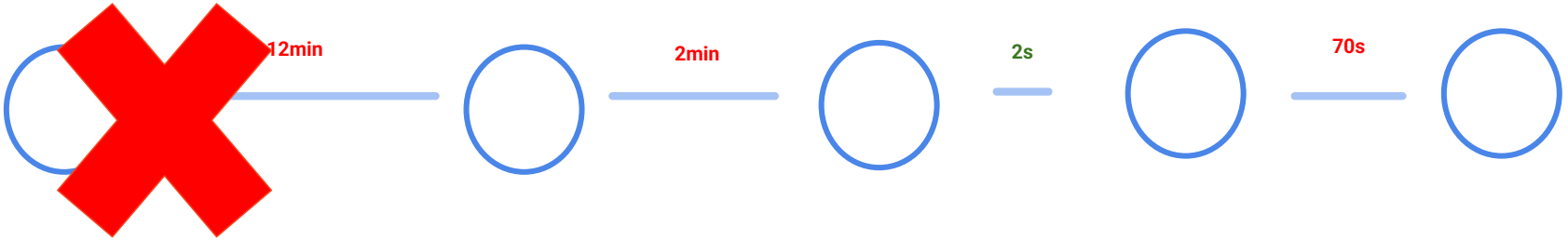

Envoy AI Gateway
Unified API
Usage Tracking
Intelligent Routing
Rate Limiting
LLM Observability
Gateway API Inference Extension
Endpoint Picker: QoS, Load and Cache Aware Routing
Deployment
Inference Service
Distributed Inference
KServe Controller
KEDA LLM Autoscaler
Model Cache Controller
GPU Scheduler/DRA
vLLM Container
Storage Container
vLLM Container
Storage Container
vLLM Container
Storage Container
vLLM Head Deployment
vLLM Worker Deployment
Model Cache / OCI
vLLM Prefill Deployment
vLLM Decoding Deployment
deepseek
Llama
Qwen
Heterogeneous GPU Farm (H100/H200/A100/MIG)
Model Registry / GCS / Azure / S3
Distributed KV Cache
LM Cache


Control Plane
api
KServe Controller
Model Cache Controller
KEDA External Scaler
Data Plane
Envoy AI Gateway
OpenAI Compatible API
Observability & Cost Tracking
LLM Rate Limiting
Intelligent Model Routing
Authn/Authz
GPU Head Node
vLLM Container
KServe Agent
Metrics Collector
Model Loading
Model Cache Agent
GPU Worker Node
...
Distributed KV Cache
Kubernetes Autoscaler (HPA)

SEP 09 – 11, 2025
CONTAINER days CONFERENCE

3 fast steps to optimize your platform

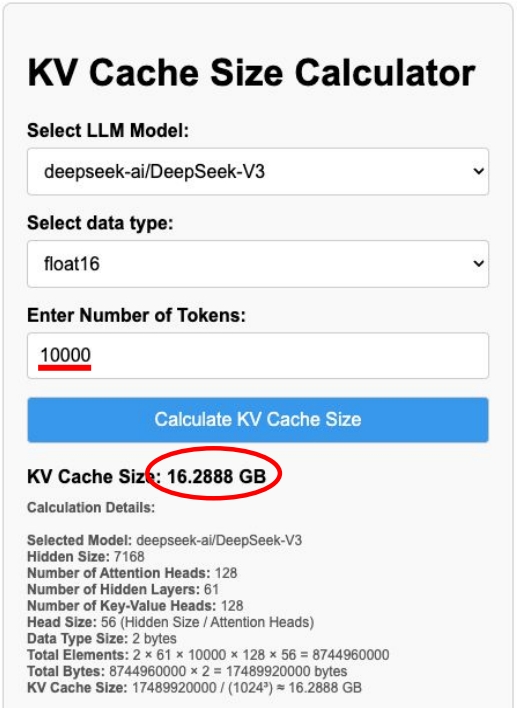
Model Caching

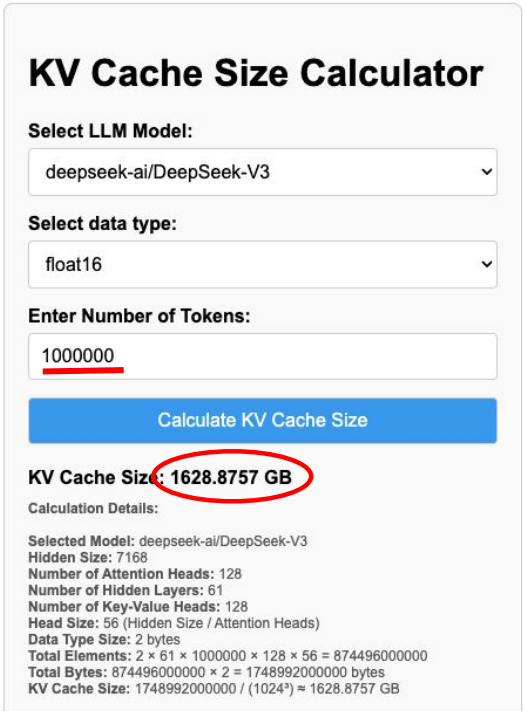


Prompt Caching

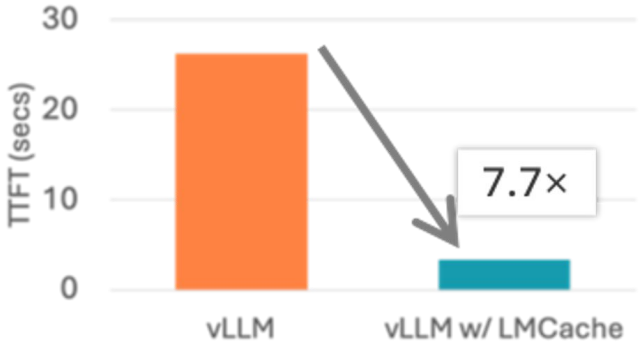
🧠 **KVCache**
Grows too fast! 📈

💡 **LMCache**
Reuses common prefixes.
✅ Faster inference!

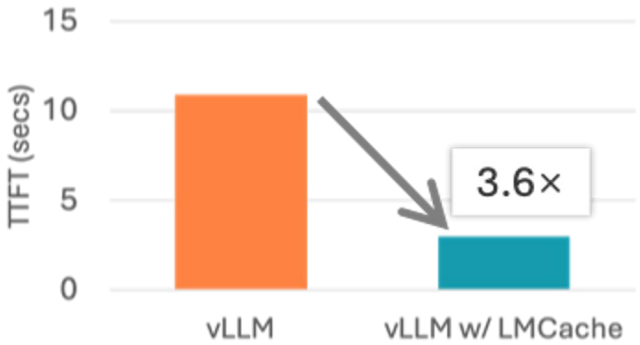




Prompt Caching



Use case 1: **long context**
Context length: **25K** tokens
(Llama 70B on A40)

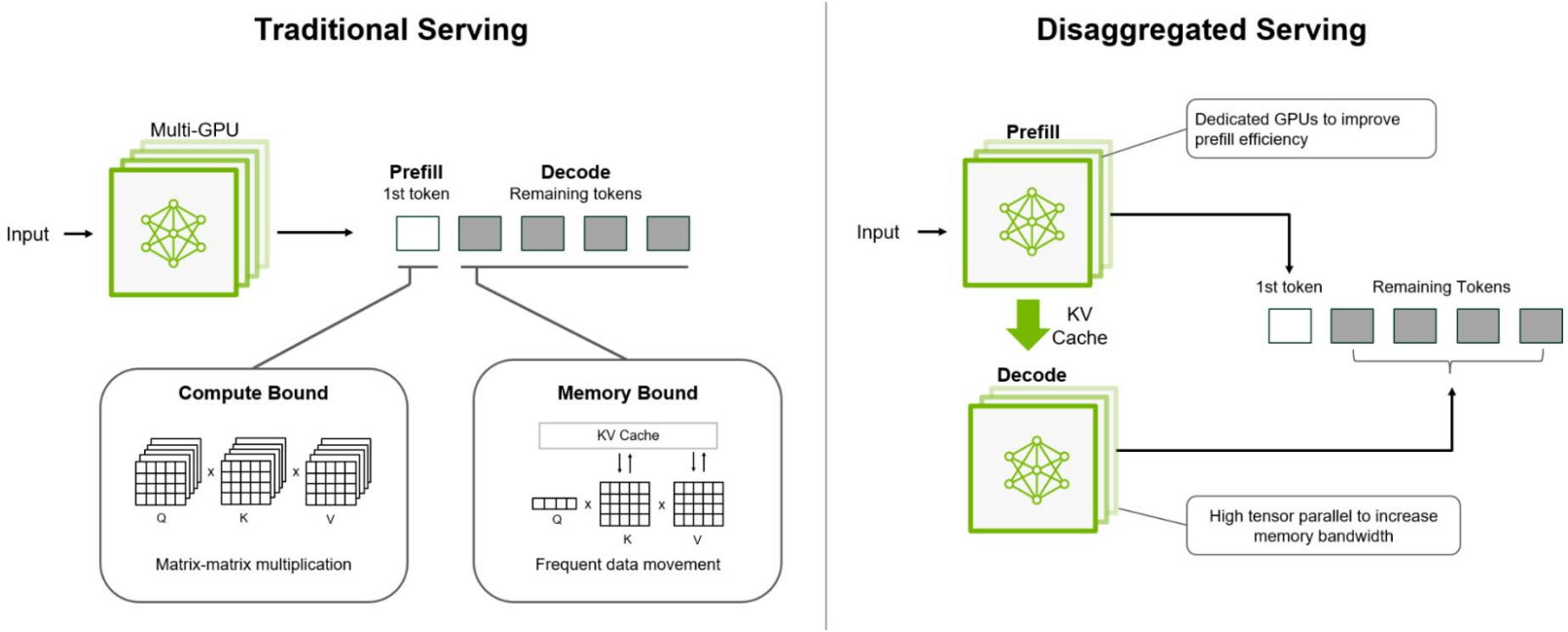


Use case 2: **RAG**
Retrieved chunks: ***4 x 2K*** tokens
(Llama 70B on A40)

LMCache drastically reduces the prefill delay (TTFT) by reusing KV caches

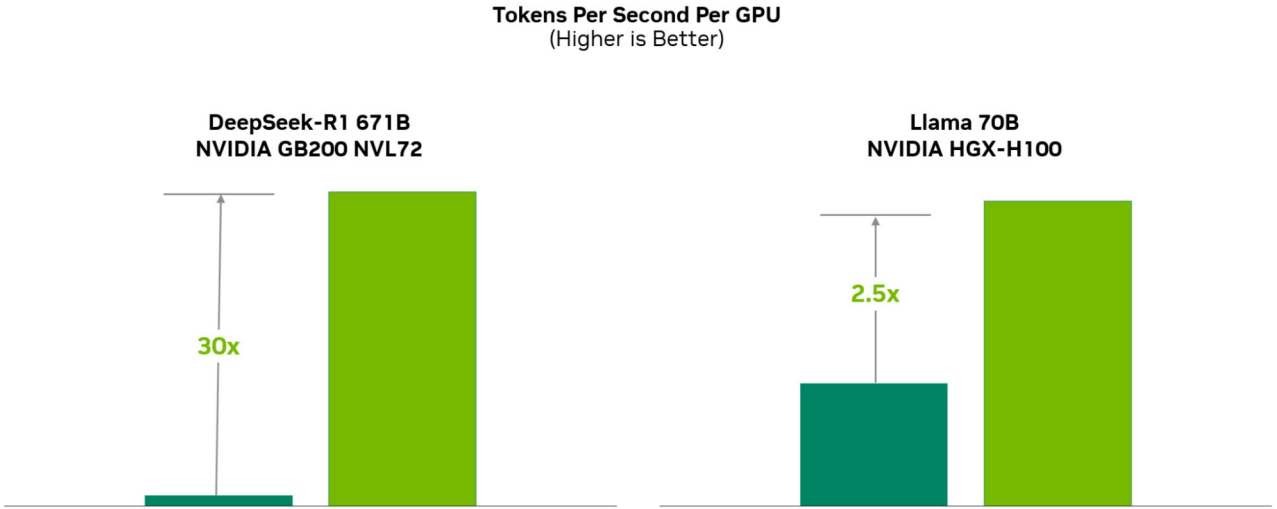
-> 3-10x delay savings
-> GPU cycle reduction in use cases like multi-round QA and RAG

Disaggregated Serving



https://developer.nvidia.com/blog/introducing-nvidia-dynamo-a-low-latency-distributed-inference-framework-for-scaling-reasoning-ai-models/

Disaggregated Serving



SEP 09 – 11, 2025

CONTAINER days CONFERENCE

Closing

Measuring the Impact of Your AI-Ready IDP

AI Serving Signals

Token Consumption

Daily processing volume

Guardrail Success

Protection execution rate

PII Leakage

Sensitive data exposure rate

AI Performance Metrics

Inference Latency

Average response time

Availability

System uptime for AI services

Throughput Gain

vs. non-optimized deployment

Platforms for LLMs Enable ISO 42001 & EU AI Act

AI-ready Internal Developer Platforms unlock compliance with emerging AI governance standards.

2023

Standard Release

First international standard for AI management systems.

100%

Governance

Complete oversight of AI assets and processes.

42001

Certification

Demonstrates commitment to responsible AI practices.

Common Pitfalls

- **Over-Engineering from Day One**
- **Underestimating Resource Requirements**
- **Neglecting Model Governance**
- **Poor Developer Adoption**

Key Takeaways

Balance Control & Flexibility

Create standardized paths that make the right way the easy way, while allowing for customization when necessary.

Start Simple, Iterate Often

Begin with core capabilities that unblock key workflows. Add sophistication based on actual user needs and feedback.

Developer Experience Drives Adoption

The most technically impressive platform is worthless if developers avoid using it. Invest in intuitive interfaces and documentation.

Efficient Resource Management is Critical

GPU utilization directly impacts both cost-effectiveness and user satisfaction. Optimization pays dividends at scale.